

Internationalisierung

Christoph Thelen

Zusammenfassung—Um eine Internationalisierung durchführen zu können, müssen zunächst die Grundlagen von Zeichen und Zeichensätzen bekannt sein. Und nur mit einer geeigneten Zeichenkodierung lassen sich Texte auch verlustfrei transferieren. Bei Altsystemen reicht es außerdem nicht, bloß die Werkzeuge auf eine neue Kodierung umzustellen. Auch die über Jahre angesammelten Datensätze müssen in die neue Kodierung überführt werden, da sonst Datenverluste vorprogrammiert sind. Neue Anforderungen werden schließlich an den bestehenden Quelltext gestellt, um das Übersetzen der Inhalte auch Nicht-Programmierern zu ermöglichen.

Index Terms—gettext, Internationalisierung, mbstring, Unicode, UTF-8, Zeichen, Zeichencode, Zeichensatz

I. EINFÜHRUNG

DIE Internationalisierung von Applikationen gehört im Zeitalter von „Web 2.0“ zum Pflichtprogramm eines Entwicklers. Dennoch entsteht in eStudy täglich neuer Code, der festverdrahtete Zeichenketten in deutscher Sprache enthält. Für eine funktionierende Internationalisierung müssen diese Zeichenketten jedoch austauschbar sein, so dass zum Beispiel ein deutscher Text durch einen englischen ersetzt werden kann. Entscheidend ist aber auch, ob die Zeichen der jeweiligen Sprache auch auf dem Bildschirm dargestellt werden können. Es nützt nichts, wenn sämtliche Texte der Anwendung auf japanisch übersetzt wurden, wenn diese Sprache nicht korrekt dargestellt werden kann.

II. ZEICHEN UND ZEICHENSÄTZE

Wie so vieles in der Informatik, wird der Begriff Zeichensatz für mehrere Dinge verwendet. Zum Beispiel ist darunter ein Repertoire von Zeichen zu verstehen, etwa „&“ oder „s“. Aber auch die Kodierung von Zeichen wird unter Zeichensatz geführt, wie bei folgender Angabe in der Kopfzeile eines Dokumentes: `Content-Type: text/html; charset=utf-8`. Es hilft daher, in anderen Kategorien zu denken und das Wort Zeichensatz nur als Oberbegriff für drei verschiedene Dinge zu nehmen [1]:

Zeichenrepertoire: Eine Auflistung der eigentlichen Zeichen. Jedes Zeichen hat dabei einen eindeutigen Namen und eine zugehörige graphische Repräsentation. Es ist hier völlig unwichtig, wie diese Daten gespeichert werden. Die Reihenfolge, in der die Zeichen dargestellt werden, spielt ebenfalls keine Rolle.

Zeichencode: Eine 1-zu-1 Abbildung von einer nicht-negativen Zahl zu einem Zeichen im Repertoire. Die Zahlen müssen dabei nicht kontinuierlich aufsteigen, sondern es können durchaus Lücken vorhanden sein. Diese Lücken werden in der Regel für spezielle Steuerzeichen verwendet.

Zeichenkodierung: Wie die einzelnen Codes in Bytes dargestellt werden können.

Der bekannteste Zeichensatz der Informatik ist wohl der ASCII-Zeichensatz. Er besteht aus 32 Steuerzeichen (Codes 0 bis 31) und 95 druckbaren Zeichen (Codes 32 bis 126). Er nimmt dadurch einen Platz von 7 Bit ein. Da ASCII beispielsweise keine Umlaute definiert hat, wurde der Zeichensatz erweitert (*Extended ASCII*). Das achte Bit wurde dafür verwendet, die fehlenden Zeichen diverser Sprachen zu ergänzen. Diese Zeichen befinden sich daher im Codebereich 127 bis 255. Der Standard ISO 8859 basiert auf einer solchen ASCII-Erweiterung, der bekannteste Standard ist ISO 8859-1 (Latin1). Microsoft wiederum hat einen eigenen Zeichensatz auf ISO 8859-1 basieren lassen: Windows 1252. Darin werden vom ISO-Zeichensatz unbenutzte Codes für zusätzliche druckbare Zeichen verwendet.

Unter der Bezeichnung Unicode ist der Standard ISO 10646 bekannt, es ist ein Repertoire für eine riesige Anzahl von Zeichen. Ebenso werden darin Zeichencodes definiert. Für Unicode existieren aber verschiedene Möglichkeiten der Kodierung [1]:

a) **UTF-32:** Jedes Zeichen wird mit 4 Bytes (32 Bits) dargestellt. Dies ist eine sehr einfache Kodierung, die allerdings nur selten verwendet wird, da der Platzbedarf für ein Zeichen meist unnötig hoch ist.

b) **UTF-16:** Hier werden die Zeichen durch 2 Bytes (16 Bit) kodiert. Das Unicode Repertoire ist dabei in mehrere Ebenen (*planes*) unterteilt. Die erste Ebene ist die *Basic Multilingual Plane* (BMP), welche die meisten der Zeichen enthält und somit auch die meisten Schriften. Insgesamt gibt es 17 Ebenen. Zeichen der BMP werden direkt mit den zwei zur Verfügung stehenden Bytes kodiert. Zeichen der übrigen Ebenen werden mit Hilfe der sog. Ersatzpaare (*surrogate pairs*) dargestellt. Dabei werden benachbarte Bytes zusammengefasst, um insgesamt mehr als 16 Bit zur Verfügung zu haben. Die Kodierung hat damit eine variable Länge.

c) **UTF-8:** Zeichen aus dem ASCII Repertoire können mit UTF-8 direkt als solche gespeichert werden. Sie benötigen auch jeweils nur 1 Byte Speicherplatz. Alle Übrigen benötigen mehr Bytes, maximal 4. Die lateinischen Buchstaben mit Diakritika benötigen zum Beispiel 2 Bytes, Zeichen aus der BMP 3 Bytes. Dadurch kann auch hier das gesamte Unicode Repertoire dargestellt werden. **Da UTF-8 sich zunehmend im Internet durchsetzt¹, sollte es als Standard für eStudy eingesetzt werden.**

Kennt ein Programm die Kodierung einer Zeichenkette, kann es die einzelnen Zeichen auslesen. Das bedeutet aber nicht, dass diese auch korrekt dargestellt werden können. Dafür benötigt das Programm zusätzliche Informationen [1]:

Der Autor kann unter der folgenden Adresse erreicht werden: christoph.thelen@mni.fh-giessen.de.

¹Quelle: Wikipedia, <http://en.wikipedia.org/wiki/UTF-8>

Eine Glyphe ist die Repräsentation einer bestimmten Form, in der ein Zeichen dargestellt werden kann. Das Zeichen *S* kann beispielsweise fett oder kursiv dargestellt werden und es wäre weiterhin das gleiche Zeichen. Es unterscheidet sich jedoch vom Zeichen *s*.

Eine Zusammenfassung mehrerer Glyphen wird als Schriftart (*font*) bezeichnet. Besitzt das Programm oder Betriebssystem nicht die richtige Schriftart um ein Zeichen darzustellen, wird anstelle des Zeichens meist ein Platzhalter eingefügt. Dies kann zum Beispiel ein Quadrat sein, oder auch eine schwarze Raute mit einem weißen „?“ in der Mitte [2].

Verwendet ein Programm dagegen eine falsche Kodierung um eine Zeichenkette auszulesen, kann es zu Problemen kommen: Zum Beispiel wenn ein als UTF-8 gespeicherter Text als Latin1 interpretiert wird, tritt der Effekt der „entstellten“ Umlaute auf.

III. VORBEREITUNG

Um die korrekte Darstellung von mit UTF-8 kodierten Zeichen zu ermöglichen, muss zunächst dem Browser die richtige Kodierung mitgeteilt werden. In der Kopfzeile einer Anfrage muss daher UTF-8 als Zeichenkodierung angegeben werden. Für den Webserver Apache kann dies über die in Listing 1 angegebenen Befehle erreicht werden.

Listing 1. Apache Konfiguration

```
AddDefaultCharset On
AddDefaultCharset utf-8
AddEncoding .atom .htm .html .xht .xhtml
        .xml .php UTF-8
```

Die Inhalte der Dateien mit den angegebenen Endungen werden damit immer mit UTF-8 markiert. Als Zweites muss die Kodierung in eStudy selbst umgestellt werden. Ein Administrator muss dazu unter **Wartung** > **Einstellungen** > **Anzeige** > **Zeichensatz** den bestehenden Wert durch UTF-8 ersetzen.

Der Browser wird jetzt die Inhalte richtig als UTF-8 erkennen, doch was ist mit PHP? Das Testskript in Listing 2 zeigt, wie PHP mit UTF-8 Zeichenketten umgeht.

Listing 2. UTF-8 Zeichenketten und PHP

```
<?php
$plain = 'Internationalization';
$utf8 = 'İntërnâtiônàlizætiøn';
echo $plain, strlen($plain);
echo $utf8, strlen($utf8);
```

Im Idealfall sollten beide Zeichenketten korrekt ausgegeben werden, außerdem sollte die Länge der Strings jeweils 20 Zeichen betragen. Der Browser kann beide Zeichenketten korrekt darstellen, allerdings sind zwei verschiedene Längen angegeben:

Der zweite String hat angeblich eine Länge von 27 Zeichen, statt den erwarteten 20! Für PHP gilt, dass ein Zeichen einem Byte entspricht. Dies ist aber bei UTF-8 nicht mehr der Fall, also muss auch hier nachgebessert werden. Die „mbstring“-Erweiterung (*MultiByte String*) bietet genau die Funktionalität, die benötigt wird.

Listing 3. Einsatz der mbstring-Erweiterung

```
<?php
$plain = 'Internationalization';
$utf8 = 'İntërnâtiônàlizætiøn';
echo $plain, mb_strlen($plain);
echo $utf8, mb_strlen($utf8);
```

Bei der Ausführung des Codes aus Listing 3 haben beide Zeichenketten jetzt auch für den PHP-Interpreter die gleiche Länge von 20. In dieser Form würde es allerdings bedeuten, dass der gesamte Quellcode von eStudy angepasst werden müsste. Glücklicherweise bietet die mbstring-Erweiterung aber eine Option an, um alle Stringfunktionen automatisch durch die Multibytevarianten zu ersetzen. Der Aufruf von `strlen()` führt damit automatisch die Funktion `mb_strlen()` aus.

Das nächste Problem tritt bei der Datenbank auf, die eStudy Tabellen müssen ebenfalls alle auf UTF-8 umgestellt werden². Das Umstellen der Tabellen ist recht einfach:

Die Strukturen der Tabellen müssen exportiert, die Zeichensätze auf UTF-8 umgestellt und schließlich die Tabellen wieder neu angelegt werden. [8] zeigt, wie dies funktioniert. Das Konvertieren der eigentlichen Daten ist der wirklich schwierige Teil:

Zuerst muss geprüft werden, in welcher Kodierung die Daten vorliegen. In der Regel ist dies „Latin1“. Dann müssen alle Daten in UTF-8 umgewandelt werden. Wenn die Daten nach der Konvertierung noch stimmen, können sie migriert werden. Aber selbst danach muss noch einmal ein Test gemacht werden, ob die jetzt eingespielten Daten auch wirklich mit der ursprünglichen Fassung übereinstimmen. Denn auch beim Wiedereinspielen können die Zeichen entstellt werden.

Wenn die gesamte Datenbank auf UTF-8 umgestellt ist, muss nur noch dafür gesorgt werden, dass neue Daten auch als UTF-8 behandelt werden. Dazu ist es am Besten, wenn direkt nach dem Verbindungsaufbau von PHP zur Datenbank der Befehl `SET NAMES 'utf8'` ausgeführt wird [6]. Damit sind alle Vorbereitungen abgeschlossen.

IV. GETTEXT

Gettext ist die GNU Internationalisierungsbibliothek. Mit ihr wurde sich bereits in [9] befasst, weswegen eine Einführung hier entfällt. Statt dessen direkt zum wichtigen Teil, wie eStudy tatsächlich übersetzt werden kann. [7] nennt fünf Dinge, die es dabei zu beachten gilt:

A. Guter sprachlicher Stil

Es soll möglichst kein Slang verwendet werden, da sonst Übersetzer, die nicht vom Fach sind, Schwierigkeiten bei der Arbeit haben werden. Auf Abkürzung soll daher ebenfalls verzichtet werden.

B. Ganze Sätze verwenden

Für einen Übersetzer ist es schwierig, eine geeignete Übersetzung zu wählen, wenn die Wörter ohne Zusammenhang auftauchen. Daher immer ganze Sätze verwenden.

²Achtung: UTF-8 wird in MySQL mit „utf8“ bezeichnet

C. Trennung erst an Absätzen

Mehrere Sätze, die zusammengehören, sollen nach Möglichkeit erst an einem Absatz getrennt werden. Dies hilft wieder, den Kontext zu bewahren.

D. Formatierte Zeichenketten anstelle von Konkatenation verwenden

Das Konkatenieren von mehreren Teilstrings und Variablen führt dazu, dass sich der entsprechende Text nicht übersetzen lässt. Statt dessen sollen die Zeichenketten über Funktionen wie etwa `sprintf` formatiert werden. Ein Übersetzer kann in der Regel mit den Platzhaltern innerhalb der Zeichenkette umgehen.

E. Vermeiden von ungewöhnlichen Steuerzeichen und Auszeichnungen

HTML-Code ist in Ordnung, aber zum Beispiel alles, was über die gängigen Entitäten hinaus geht, soll vermieden werden.

Viele weitere Regeln finden sich im Handbuch von gettext [7].

V. FAZIT

Die Internationalisierung von eStudy ist kein leichtes Unterfangen. Besonders beim Umwandeln der Datenbank ist höchste Vorsicht geboten. Nur all zu leicht können hier Daten verloren gehen. Ein methodisches Vorgehen und geeignete Testfälle sind unerlässlich. Das Bearbeiten der Quellen wird dagegen eher eine Fleißarbeit. Aber auch dies muss getan werden, damit das Übersetzen der Texte möglichst einfach gestaltet werden kann. Mindestens die fünf Regeln aus dem vorhergehenden Abschnitt sollen eingehalten werden.

LITERATUR

- [1] KORPELA, JUKKA: A tutorial on character code issues. Online im Internet: URL: <http://www.cs.tut.fi/~jkorpela/chars.html> - Stand: 26.09.2008
- [2] SPOLSKY, JOEL: The Absolute Minimum Every Software Developer Absolutely, Positively Must Know About Unicode and Character Sets (No Excuses!). Online im Internet: URL: <http://www.joelonsoftware.com/articles/Unicode.html> - Stand: 26.09.2008
- [3] SUGALSKI, DAN: What the heck is: A string. Online im Internet: URL: <http://www.sidhe.org/~dan/blog/archives/000255.html> - Stand: 26.09.2008
- [4] BRAY, TIM: Characters vs. Bytes. Online im Internet: URL: <http://www.tbray.org/ongoing/When/200x/2003/04/26/UTF> - Stand: 26.09.2008
- [5] BRAY, TIM: On the Goodness of Unicode. Online im Internet: URL: <http://www.tbray.org/ongoing/When/200x/2003/04/06/Unicode> - Stand: 26.09.2008
- [6] EIBECK, FLORIAN: Multilingual Websites with PHP. Online im Internet: URL: <http://blog.thinkphp.de/archives/342-Multilingual-Websites-with-PHP.html> - Stand: 26.09.2008
- [7] FREE SOFTWARE FOUNDATION: GNU gettext utilities. Online im Internet: URL: <http://www.gnu.org/software/gettext/manual/gettext.html> - Stand: 26.09.2008
- [8] STRIEGEL, JASON: MySQL database migration: latin1 to utf8 conversion. Online im Internet: URL: http://www.hackszine.com/blog/archive/2007/05/mysql_database_migration_latin.html - Stand: 26.09.2008
- [9] ELLER, STEFAN: Internationalisierung von Webseiten. Online im Internet: URL: http://wiki.mni.fh-giessen.de/index.php/Internationalisierung_von_Webseiten - Stand: 26.09.2008

Christoph Thelen ist Student der Fachhochschule Gießen-Friedberg am Fachbereich Mathematik, Naturwissenschaften und Informatik. Seit 2007 ist er Entwickler und Administrator der Lern- und Kollaborationsplattform eStudy. Neben der Architektur von (Web-)Applikationen interessiert er sich außerdem für agile Entwicklungsmethoden wie Extreme Programming und Scrum.